КОНСТРУКТОРИ

доц. д-р Нора Ангелова

Жизнен цикъл на обект

- Създаване на обект в даден скоуп(област) – заделя се памет и член-данните могат да се инициализират
- Достига се до края на скоупа(област)
- Обектът и паметта, която е асоциирана с него се разрушава

Жизнен цикъл на обект

- **Създаване на обект в даден скоуп(област) – заделя се памет и член-данните могат да се инициализират**
- Достига се до края на скоупа(област)
- Обектът и паметта, която е асоциирана с него се разрушава

Конструктори

- Как се създават обектите?
- Кой заделя паметта?
- Може ли да променим тази част от жизнения цикъл на един обект?

КОНСТРУКТОРИ

- Конструкторите са член-функции, чрез които се инициализират **член-данните** на обект от даден клас(структура)
- Това се осъществява при създаването на обект

КОНСТРУКТОРИ

- Конструктор може да се създаде автоматично (по подразбиране)
- Възможно е да се дефинират множество конструктори(с параметри и без параметри)

КОНСТРУКТОРИ

Специфики

- **В клас не е дефиниран конструктор**

```
<име_на_клас> <обект>;
```

Как създавахме обектите до момента?!

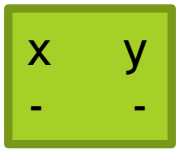
Автоматично в класа се създава подразбиращ се конструктор(конструктор без параметри) и инициализацията на обекта се осъществява чрез него.

Този конструктор изпълнява редица действия, като заделяне на памет за обектите, инициализиране на някои системни променливи и др.

КОНСТРУКТОРИ

- **В клас не е дефиниран конструктор**

```
class Point2D {
  public:
    void print() const;
  private:
    double x;
    double y;
};
...
int main() {
   Point2D myFirstPoint; // Инициализира се чрез
                         // подразбиращия се конструктор,
                         // създаден от компилатора
   myFirstPoint.print(); // Какъв е резултатът?
   return 0;
}
```



Конструктори

- Създаване на конструктори
  - Името на конструктора съвпада с името на класа (структурата)
  - Не се указва тип на връщания резултат
  - Връща референция към създадения обект – this
  - Изпълнява се автоматично при създаването на обекти
  - Не може да се извика явно (error: ~~obj.<конструктор>~~)

КОНСТРУКТОРИ

Специфики

- **В клас явно е дефиниран конструктор/и**
  - Конструктор без параметри (подразбиращ се конструктор)
  - Конструктори с параметри

* В класа вече има дефинирани конструктори.
Подразбиращ се конструктор **НЕ** се създава автоматично.

КОНСТРУКТОРИ

Специфики

- **В клас явно е дефиниран конструктор/и**

Дефиницията на обект от този клас трябва задължително да е в съответствие с един от съдържащите се в класа конструктори.

* В класа вече има дефинирани конструктори.
Подразбиращ се конструктор **НЕ** се създава автоматично.

Конструктори

- Инициализация на член-данните

КОНСТРУКТОРИ

- **В клас явно е дефиниран конструктор по подразбиране (default-ен конструктор)**
  - Няма параметри
  - Позволява създаването на обекти без подаване на инициализиращи стойности
  - Инициализира член-данните с подразбиращи се/начални стойности

```
class Point2D {
  public:
    Point2D(); // Конструктор по подразбиране
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D() {
    x = 0; // Инициализира x с 0
    y = 0; // Инициализира y с 0
}
```

КОНСТРУКТОРИ

Специфики

- **В клас явно е дефиниран конструктор/и**
  - Параметрите представляват стойности, с които член-данните се инициализират
  - Не заместват конструктора по подразбиране

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue, double yValue) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}

int main() {
   Point2D myFirstPoint; // Предизвиква грешка
                         // Няма конструктор по подразбиране
   return 0;
}
```

КОНСТРУКТОРИ

Специфики

- **В клас явно е дефиниран конструктор/и**
  - Параметрите представляват стойности, с които член-данните се инициализират
  - Не заместват конструктора по подразбиране

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue, double yValue) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}

int main() {
   Point2D myFirstPoint(3.5, 4.2); // Създава обект и инициализира
                                   // x с 3.5 и y с 4.2
   return 0;
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - В тялото на конструктора

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue, double yValue) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - В заглавието на конструктора – обобщена синтактична конструкция

    Обобщената синтактична конструкция инициализира член-данните в заглавието преди изпълнението на тялото на конструктора.

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};
// Инициализира член-данните x и y
Point2D::Point2D(double xValue, double yValue) : x(xValue), y(yValue)
{}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - Комбинация на двата подхода

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};

// Инициализира член-данната x
Point2D::Point2D(double xValue, double yValue) : x(xValue) {
    y = yValue; // Инициализира y
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - В заглавието на конструктора – обобщена синтактична конструкция

    Защо съществува тази стъпка преди изпълнение на тялото на конструктора?

```
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};
// Инициализира член-данните x и y
Point2D::Point2D(double xValue, double yValue) : x(xValue), y(yValue)
{}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - Член-данни на клас, които са обекти
    В дефиницията на конструктора на класа се използват конструкторите на класовете на обектите

```
class Rect {
   public:
     Rect(double, double, double, double);
     ...
   private:
     Point2D topLeft;
     Point2D bottomRight;
};

Rect::Rect(double x1, double y1, double x2, double y2) {
    topLeft = Point2D(x1, y1);
    bottomRight = Point2D(x2, y2);
}
```

Какво се случва в действителност?

КОНСТРУКТОРИ

- Инициализация на член-данните
  - Член-данни на клас, които са обекти

  1. Автоматично се извикват конструкторите по подразбиране на всички член-данни, които са обекти (АКО ТОВА Е ВЪЗМОЖНО – такъв е дефиниран или може да се създаде автоматично)
  2. Член-данните се инициализират повторно в тялото на конструктора

```
class Rect {
  public:
    Rect(double, double, double, double);
    ...
  private:
    Point2D topLeft;
    Point2D bottomRight;
};
Rect::Rect(double x1, double y1, double x2, double y2) {
    topLeft = Point2D(x1, y1);
    bottomRight = Point2D(x2, y2);
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните
  - Член-данни на клас, които са обекти

Решение(извикване на определен конструктор точно веднъж):

```
class Rect {
  public:
    Rect(double, double, double, double);
    ...
  private:
    Point2D topLeft;
    Point2D bottomRight;
};
// Извикват се само конструкторите с два параметъра
Rect::Rect(double x1, double y1, double x2, double y2) :
topLeft(x1, y1), bottomRight(x2, y2)
{}
```

Функции и конструктори

- Подразбиращи се параметри

Задаването на подразбираща се стойност се извършва чрез задаване на конкретна стойност в прототипа на функцията или в заглавието на нейната дефиниция.

```
(Вариант 1)
class Point2D {
  public:
    Point2D(double xValue = 0, double yValue = 0);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue, double yValue) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}
```

Функции и конструктори

- Подразбиращи се параметри

Задаването на подразбираща се стойност се извършва чрез задаване на конкретна стойност в прототипа на функцията или в заглавието на нейната дефиниция.

```
(Вариант 2)
class Point2D {
  public:
    Point2D(double xValue, double yValue);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue = 0, double yValue = 0) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}
```

Функции и конструктори

- Подразбиращи се параметри

Ако параметър на функция е подразбиращ се, всички параметри след него също трябва да са подразбиращи се.

Подразбиращите се параметри могат да се използват за функции и конструктори.

Функции и конструктори

- Подразбиращи се параметри

Колко конструктора са дефинирани?

```
class Point2D {
  public:
    Point2D(double xValue = 0, double yValue = 0);
    void print() const;
  private:
    double x;
    double y;
};

Point2D::Point2D(double xValue, double yValue) {
    x = xValue; // Инициализира x
    y = yValue; // Инициализира y
}
```

ВРЕМЕ ЗА ВАШИТЕ ВЪПРОСИ